International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.6, pp : 826-829 1 June 2014

Effective and Efficient Optimization in RC4 Stream

N.Sivasankari [1],A.Yogananth [2]
Sembodai Rukmani Varatharajan Enginnering College,Sembodai.Tamilnadu.
Sivasankari.sec@gmail

Abstract—At present technology, power saving technique is one of the important criteria while designing a hardware for RC4 stream cipher. So this paper proposes a BCD to Excess- 3-adder to build an efficient hardware with parallel processing system for low power consumption without compromising the speed and security in cryptography. Loop unrolling and Pipeline concepts are used for expeditious hardware implementation. This design is realized on XC3S100E FPGA with VHDL language.

KEY WORDS— BCD to Excess -3 -adder, Cryptography, Loop unrolling, Pipelining, RC4, Stream cipher.

I. Introduction

RC4 is developed by Ron Rivest in 1987 for RSA (Rivest-Shamir-Adleman) data security in the field of cryptography. Still it is most popular in many network protocols Such as SSL (Secure Socket Layer), TLS (Transport Layer Security) for communication security over internet due to its simplicity and ease of implementation. Stream cipher uses a simple method to generate a cipher text by combining the plain text with secret key by Exclusive OR operation .The security of stream cipher depends on the quality of the algorithm. Whenever the browser establishes a secure communication to a Web site, RC4 stream cipher is used. It also provides higher efficiency and speed than block cipher. The hardware implementation of cryptographic algorithm plays a significant role than software because of its real time high speed and security .The throughput of the hardware implementation of proposed design was N* clock frequency. Where N is the number of bits produced in every clock cycle.Hardware implementation of RC4 combine the idea of loop unrolling and pipeline that produces 2RC4 key stream bytes per Clock cycle. But in this power consumption and required area for this design are large.

II. Material and Methodology

1. RC4 Algorithm

It uses S-box S, an array of length N, where each location of S stores 1 byte (typically, N=256). A secret Key k of size *l* bytes is used to scramble this permutation (typically, $5 \leq l \leq 16$).Array K of length N holds the main key, with secret key k repeated as K[y*]* =k [y mod *l]*, for $0 \leq y \leq$ N-1. At the transmitting end, cipher text is generated by C=M EX-OR Z, and deciphering at the receiving end by M = C EX-OR Z. RC4 has two algorithms, namely KSA (Key Scheduling Algorithm) and PRGA (Pseudo-Random Generation Algorithm).

1.1 KSA Algorithm:

KSA is used to initialize the permutation in the array "S".
for i from 0 to 255.
S [i]:= i
endfor
j :=0
For i from o to 255
j :=(j + S [i] +key [I mod key length]) mod 256
Swap (S [i], S [j]
Endfor

1.2 PRGA Algorithm:

PRGA begin with array S that was swapped in the KSA.
i: =0
j :=0
While generating output:
i:= (i+1) mod 256
j:= (j + s [i]) mod 256 Swap (s [i] ,s [j])
Output s [(s [i] + s [j] mod 256]
Endwhile

2. EXISTING SYSTEM

In "Design 1" RC4 hardware provides a key stream throughput of 1 byte-per-cycle using the idea of loop unrolling .and in "Design2" which provides a key stream throughput 2 byte –per-cycle by combining the hardware pipeline and loop unrolling.

2.1. Loop unrolling method for design1:

In this, we take the two consecutive values of Z together for the consecutive plaintext bytes to be encrypted. Assume the initial values of variables i_0, j_0, S_0 respectively. After the first execution of PRGA loop, the values will be i_1, j_1 S_1.

(A) To compute i_1, i_2:
The value i_0 is incremented into 1 and 2 by the same clock pulse. So the synchronous counter i_1 and i_2 loaded with 00000001, 00000010. This is for the first two rounds of RC4 in both KSA and PRGA.

TABLE-1: Two consecutive Loops of RC4 Stream Cipher

First loop	Second loop
$i_1 = i_0 + 1$	$i_2 = i_1 + 1 = i_0 + 2$
$j_1 = j_0 + S_0[i_1]$	$j_2 = j_1 + S_1[i_2] = j_0 + S_0[i_1] + S_1[i_2]$
Swap $S_0 [i_1]$, $S_0 [j_1]$	Swap $S_1 [i_2]$, $S_1 [j_2]$
$Z_1 = S_1[S_0 [i1] + S_0[j_1]]$	$Z_2 = S_2 [S_1[i_2] + S_1[j_2]]$

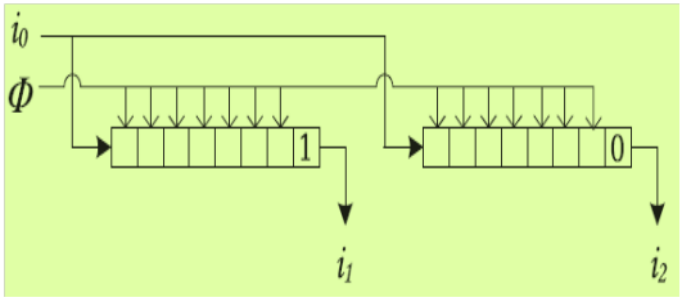


Fig-1: To Compute i_1, i_2 .[circuit 1]

(B) To compute j_1, j_2:

Two 8 bit registers are used to store the computed value j_1, j_2.for j_2 calculation, here two cases are considered. In this concept all are modulo 256 additions.

$$j_2 = j_0 + S_0[i_1] + S_1[i_2] = \begin{cases} j_0 + S_0[i_1] + S_0[i_2] \text{ if } i_2 \neq j_1, \\ j_0 + S_0[i_1] + S_0[i_1] \text{ if } i_2 = j_1. \end{cases}$$

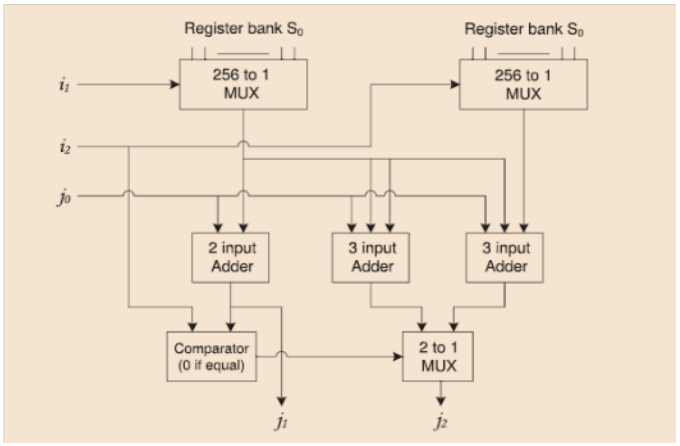


FIG-2: To Compute j_1, j_2[circuit 2]

(C) Swapping Operation:

In table-1, a third row finds any one of the given swap possibilities. The required conditions are selected by 8:1 multiplexer. If variables are two, there is a eight possible combination of swapping. It raises the secrecy level of cipher text to transmit.

Table-2: Register to Register Transfer in Swap operation

No	Condition	Register-to-Register Transfer
1	$i_2 \neq j_1 \& j_2 \neq i_1 \& j_2 \neq j_1$	$S_0[i_1] \rightarrow S_0[j_1], S_0[j_1] \rightarrow S_0[i_1], S_0[i_2] \rightarrow S_0[j_2], S_0[j_2] \rightarrow S_0[i_2]$
2	$i_2 \neq j_1 \& j_2 \neq i_1 \& j_2 = j_1$	$S_0[i_1] \rightarrow S_0[i_2], S_0[i_2] \rightarrow S_0[j_1] = S_0[j_2], S_0[j_1] \rightarrow S_0[i_1]$
3	$i_2 \neq j_1 \& j_2 = i_1 \& j_2 \neq j_1$	$S_0[i_1] \rightarrow S_0[j_1], S_0[i_2] \rightarrow S_0[i_1] = S_0[j_2], S_0[j_1] \rightarrow S_0[i_2]$
4	$i_2 \neq j_1 \& j_2 = i_1 \& j_2 = j_1$	$S_0[i_1] \rightarrow S_0[i_2], S_0[i_2] \rightarrow S_0[i_1] = S_0[j_1] = S_0[j_2]$
5	$i_2 = j_1 \& j_2 \neq i_1 \& j_2 \neq j_1$	$S_0[i_1] \rightarrow S_0[j_2], S_0[j_2] \rightarrow S_0[j_1] = S_0[i_2], S_0[j_1] \rightarrow S_0[i_1]$
6	$i_2 = j_1 \& j_2 \neq i_1 \& j_2 = j_1$	$S_0[i_1] \rightarrow S_0[j_1] = S_0[i_2] = S_0[j_2], S_0[j_1] \rightarrow S_0[i_1]$
7	$i_2 = j_1 \& j_2 = i_1 \& j_2 \neq j_1$	Identity permutation, no data transfer
8	$i_2 = j_1 \& j_2 = i_1 \& j_2 = j_1$	Impossible, as it implies $i_1 = j_1 = i_1 + 1$

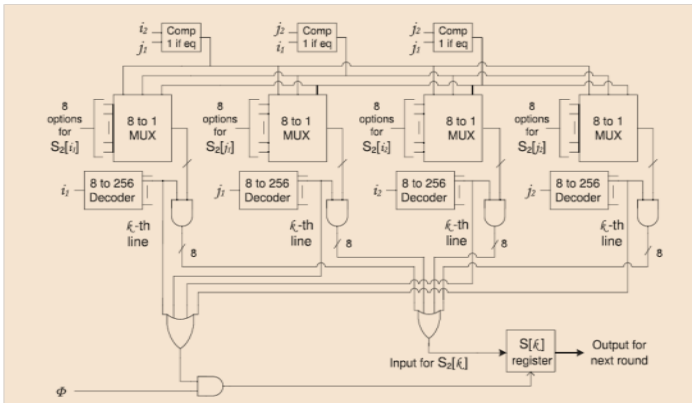


FIG-3: Circuit to Swap[circuit 3]

(D) Calculation of Z_1, Z_2:

Once the key setup phase is completed the second phase is the pseudorandom number generator (PRGN).By this loop unrolling RC4 can completely bypass the generation of S_1 and move directly from S_0 to S_2. For Z_1,addition of two value $S[i_1]$ and $S[j_1]$ are performed by 2 input parallel adder.

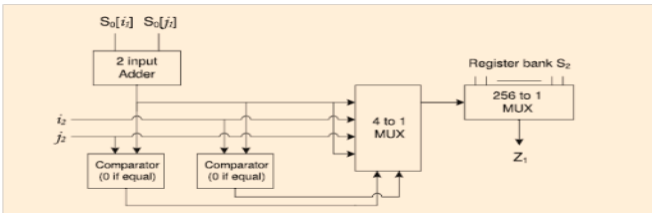


FIG-4: To Compute Z_1[circuit 4]

In the case of Z2, we unwrap one cycle of RC4 and gather the values of S $[i_2]$ and $S[j_2]$ from the S_0 state.$S_1[i_2]$ and $S_1[j_2]$ receives the values from the appropriate registers of S0.An 8 to 1 MUX unit is controlled by the outputs of three comparators comparing 1) i_2 and j_1, 2) j_2 and i_1, 3) j_2 and j_1.

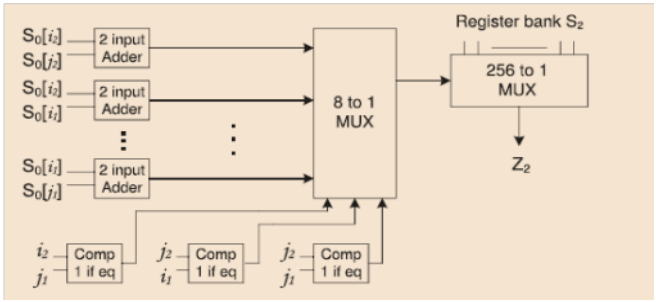


FIG-5:To compute Z_2[circuit 5]

2.2 Design 1 complete architecture:

It produces 2N bytes of output stream in N iteration , over 2N clock cycle. Let the stage of the PRGA circuit shown in Fig 6 be the nth stage. This actually denotes the nth iteration of our model, which produces the output bytes Zn+1 and Zn+2. The initial

Clock pulse ø0 is an extra one, and the production of the output Bytes lag the feedback cycle by one clock pulse in every

Iteration (e.g., øn+3 in case of nth iteration).

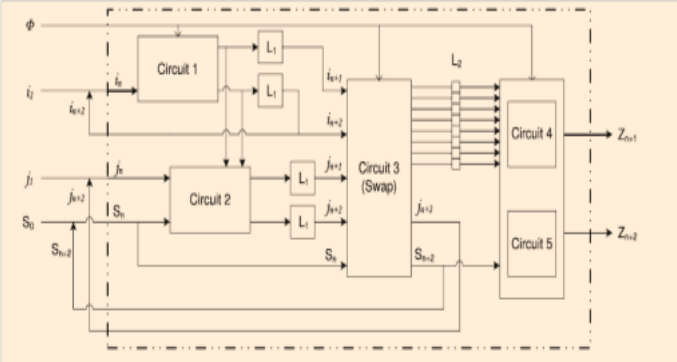


FIG-6:PRGA Stage of Design 1

2.3. Pipeline Design for 2 byte/Clock

To get maximum throughput/speed two designs of 2 stage hardware pipeline and loop unrolling are fused in this methodology. This is obtained for case with 2-stage PRGA pipeline and KSA circuit with double iteration per cycle in each case. In this case 128 cycles for S-box preparation is needed at the KSA stage, and then onward after a gap of one cycle,2 bytes per cycle are generated for encryption .

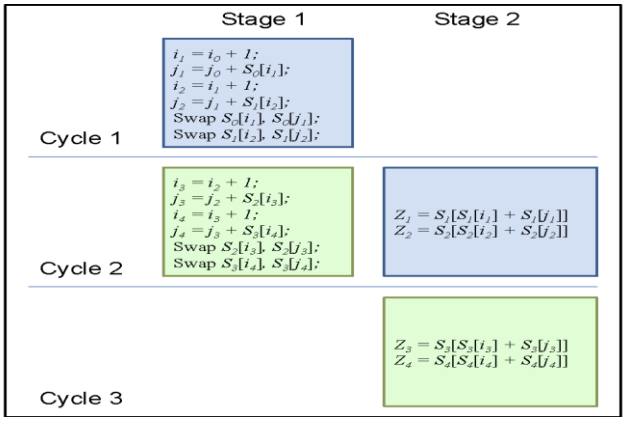


FIG-7: Pipeline structure of design 2

3. MODIFICATION

3.1 Modified Carry Select Adder (MCSA) scheme:

As to replace the N bit RCA in design1 and design 2, an N+1 bit BEC is used, so in the given designed architecture of MCSA, the 4-bit RCA is used in each block and thus the BEC used is of 5-bit wide. In the similar way MCSA architectures are designed for 8-bit, 16-bit, 32-bit and 64-bit. Conventional CSA is still area-consuming due to the dual ripple carry adder structure. The metrics for measurement of performance of a circuit are area, power and delay. So, after designing MCSA for 8-bit, 16-bit, 32-bit and 64-bit its area, power and delay are analyzed. The results so obtained are then compared with the results of conventional CSA. Modified Carry Select-Adder (MCSA) design is proposed, which make use of single RCA and Binary to Excess-1 Converter (BEC) instead of using dual RCAs to reduce area and power consumption with small speed penalty. As the base of proposed design is that the number of logic gates used in BEC is less than that of RCA. Thus BEC replaces the RCA with Cin=1 instead of using dual RCAs to reduce area and power consumption of the conventional CSA. To replace the N-bit RCA, an N+1 bit BEC is required. The MCSA architecture for 16-bit is shown in Figure 8. The importance of BEC logic comes from the large silicon area reduction when designing MCSA for large number of bits.

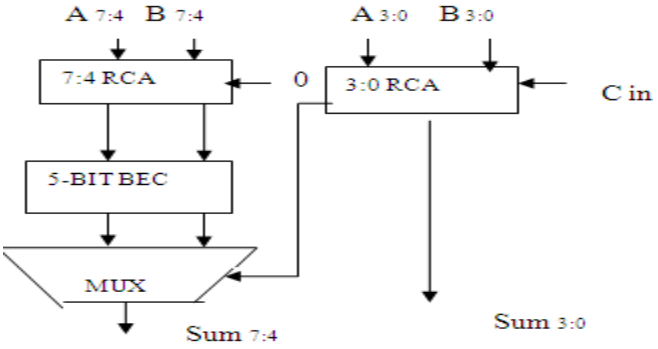


FIG -8: 4 Modified Carry Select Adder

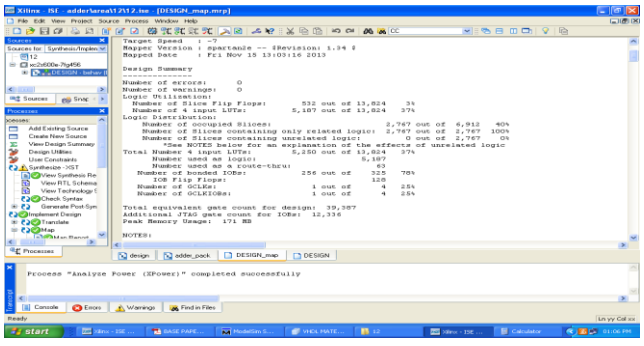
3. 2.Comparison of Modified Design 1 and Design 2

Operation	Design 1	Design 2
Per KSA round	1	1
Complete KSA	256+1=257	256+1=257
N Byte of PRGA	256+2	256/2+2
N Bytes of RC4	257+(256+2)=256+259	257+(256/2+2)=256/2
Cycles per byte	1+(259/256)	½+(259/256)

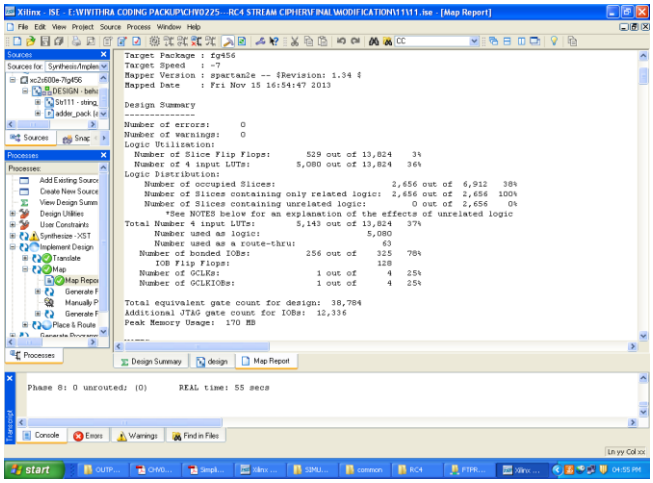
III. Results and Outputs

For simulation purpose Modelsim simulator is used and to synthesis Xilinx ISE is employed. For hardware the Spartan-3E family is a superior alternative to mask programmed ASICs FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs By Xilinx Corp.

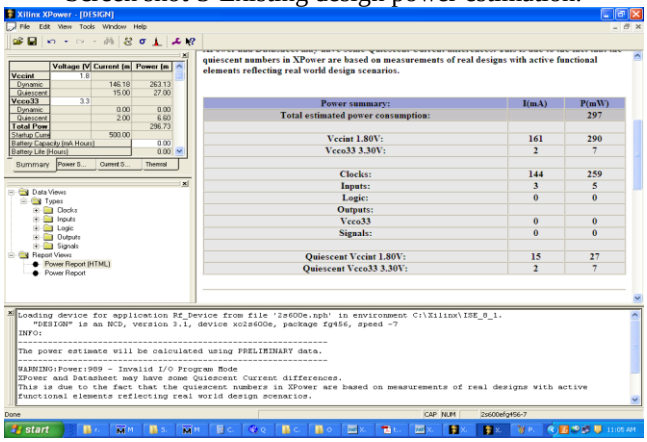
Screen shot-(1)–Shows Existing design required Area.



Screen shot-2- Modified design required Area.

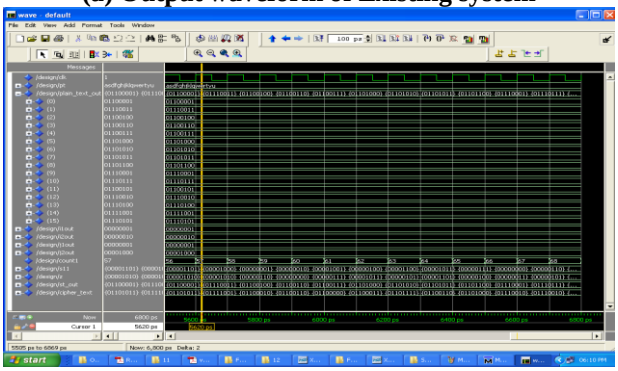


Screen shot-3-Existing design power estimation.

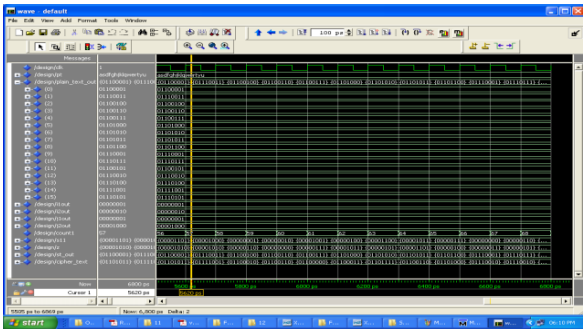


Screen shot-4-Modified design Power estimation.

(a) Output waveform of Existing system



(b)Output Waveform of modified design



IV.CONCLUSION

Our second design tops the first one by combining the idea of loop unrolling with that of efficient hardware pipelining to obtain two key stream bytes per cycle. This is the fastest known RC4 hardware architecture till date. Instead of RCA, a modified carry select adder is implemented by using BEC circuit which is used to reduce area, power and delay .This project is examined for secure communication by giving plain text from the switches and received cipher text as the LED display. Programming upon the FPGA, hopefully we can see the message succeed, to ensure this project has been successfully completed. The suggested RC4 cipher can adopt modification in future to improve their secrecy in future.

References

i. *Software Performance Results from the eSTREAM Project, eSTREAM, the ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/perf/#results, 2012.nt'l Workshop Selected Areas in Cryptography (SAC '01), vol. 2259, pp. 1-24, 2001.*

ii. *Akhilesh Tyagi, "A Reduced Area Scheme for Carry-Select Adders", IEEE International Conference on Computer design, pp.255-258, Sept 1990.*

iii. *T.-Y. Chang and M.-J. Hsiao,"Carry-Select Adder using single Ripple-Carry Adder", Electronics letters, vol.34, pp.2101-2103, October 1998.*

iv. *Deepthi Obul Reddy, P.Ramesh Yadav" Carry Select Adder with Low Power and Area Efficiency" International Journal of Engineering Research and Development e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com Volume 3, Issue 3 (August 2012), PP. 29-35.*

v. *S.R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," Proc. Eighth Ann. Int'l Workshop Selected Areas in Cryptography (SAC '01), vol. 2259, pp. 1-24, 2001.*

vi. *(vi).D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C.E. Goutis, "Comparison of the Hardware Implementation of Stream Ciphers," Int'l Arab J. Information Technology, vol. 2, no. 4, pp. 267- 274, 2005.*

vii. *Kitsos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, "Hardware Implementation of the RC4 Stream Cipher," Proc. IEEE 46th Midwest Symp. Circuits and Systems, http:// dsmc.eap.gr/en/members/pkitsos/papers/Kitsos_c14.pdf, 2003.*

viii. *D.P. Matthews Jr., "System and Method for a Fast Hardware Implementation of RC4," US Patent Number 6549622, Campbell, CA, http://www.freepatentsonline.com /6549622.html, Apr. 2003.*

ix. *D.P. Matthews Jr., "Methods and Apparatus for Accelerating ARC4 Processing," US Patent Number 7403615, Morgan Hill, CA,http://www.freepatentsonline.com /7403615.html, July 2008.*

x. *Santhoshkannan.C, J.Banumathi," Vlsi realisation of fast carry adderinbinary excess" International Journal of Communications and Engineering Volume 04– No.4, Issue: 02 March2012*